Proyecto Final – Desarrollo de una API REST con MongoDB (TypeScript)

Objetivo

Desarrollar una **API REST** utilizando **Node.js, Express, MongoDB y TypeScript**, aplicando una arquitectura **MVC**, autenticación con **JWT**, validaciones con **Zod** y operaciones **CRUD completas**, cumpliendo con los requisitos detallados a continuación.

Requisitos Generales

1. Entidad a Administrar

Elegir **una entidad principal** para gestionar dentro de la API. Algunos ejemplos (a modo orientativo): - Productos - Usuarios - Tareas - Pedidos - Libros - Preguntas de trivia

> La elección es libre, pero debe justificarse y mantenerse coherente en toda la aplicación.

2. Definición del Esquema de Datos (Mongoose)

- Utilizar **Mongoose** para definir el esquema de la entidad.
- Incluir:
  - Campos relevantes según la entidad.
  - Tipos de datos correctos.
  - Validaciones (required, min/max, unique, etc.).
  - Restricciones y valores por defecto cuando corresponda.

3. Estructura del Proyecto (Arquitectura MVC)

El proyecto debe respetar una estructura clara basada en **MVC**:

- **Model**
  - Definición del esquema y modelo con Mongoose.
- **Controller**
  - Implementación de las funciones CRUD:
    - Crear
    - Leer (uno y todos)
    - Actualizar
    - Eliminar
- **Routes**
  - Rutas RESTful bien definidas para cada operación CRUD.
  - Separación clara de responsabilidades.

4. Middleware de Autenticación (JWT)

- Implementar autenticación utilizando **JSON Web Tokens (JWT)**.
- Incluir:
  - Endpoint de login (y registro si la entidad lo requiere).
  - Generación y validación del token.
  - Middleware para proteger rutas sensibles.

5. Validación de Datos de Entrada

- Utilizar la librería **Zod** para validar los datos recibidos en requests.
- Aplicar validaciones en:
  - Creación de recursos.
  - Actualización de recursos.
- Manejar correctamente los errores de validación, devolviendo respuestas claras y apropiadas.

6. Base de Datos

- Utilizar **MongoDB** como base de datos.

- Implementar correctamente todas las operaciones **CRUD** sobre la colección correspondiente.

7. Endpoint de Agregación

Crear **al menos un endpoint adicional** que utilice **aggregations de MongoDB**, adaptado a la entidad elegida.

Ejemplos: - **Pedidos**: calcular el total de ventas o ventas por mes. - **Usuarios**: promedio de edad, cantidad de usuarios activos. - **Productos**: precio promedio, stock total.

> El endpoint debe tener sentido dentro del contexto del proyecto.

Requisitos Técnicos Obligatorios

- La **API debe estar desarrollada en TypeScript**.
- El proyecto debe estar correctamente tipado.
- Uso de buenas prácticas de código y organización.

Alternativas de Entrega del Proyecto Final

El proyecto puede entregarse de cualquiera de las siguientes formas:

Opción 1

Continuar con un **trabajo entregado de forma parcial**, siempre y cuando: - Se valide que **cumple con todos los requisitos de esta consigna**. - Se completen y corrijan los puntos faltantes si los hubiera.

Opción 2

Tomar como referencia los proyectos desarrollados en clase, **personalizándolos** y asegurando una funcionalidad completa:

- https://github.com/GabrielAlberini/mongo-frontend
- https://github.com/GabrielAlberini/mongo-backend

Se espera una adaptación propia del estudiante, no una copia directa.

Frontend

- **No es obligatorio** desarrollar un frontend.
- El **repositorio obligatorio es únicamente el de la API**.
- Quien desee sumar un frontend puede hacerlo, pero:
    - No suma puntaje adicional.
    - Se evaluará exclusivamente la API.

Deploy

- **No es obligatorio** que la API esté en producción.
- De manera opcional, puede deployarse en **Render** u otra plataforma similar.

Entregables

1. **Repositorio Git** con el código fuente de la API.
2. Archivo **README.md** que incluya:
    - Descripción del proyecto.
    - Requisitos previos.
    - Pasos para instalar dependencias.
    - Variables de entorno necesarias.
    - Comando para ejecutar el proyecto.
3. **Documentación mínima de la API**, incluyendo:
    - Listado de endpoints.
    - Métodos HTTP.
    - Ejemplos de request y response.

Criterios de Evaluación

- Cumplimiento completo de la consigna.
- Correcta implementación de MVC.
- Uso adecuado de JWT y Zod.
- Buenas prácticas y orden del código.
- Correcto uso de TypeScript.
- Coherencia y funcionalidad general de la API.